A Scalable Architecture for Providing Deterministic Guarantees

SRINIVAS VUTUKURY
vutukury@cse.ucsc.edu
Computer Sciences Department
University of California
Santa Cruz, CA 95064

J.J. GARCIA-LUNA-ACEVES
jj@cse.ucsc.edu
Computer Engineering Department
University of California
Santa Cruz, California 95064
Networking and Security Center
Sun Microsystems Laboratories
Palo Alto, California 94303

Abstract—**The Internet community has proposed the Integrated Services architecture (Intserv) and the signaling protocol RSVP to provide deterministic guarantees (bandwidth, delay and jitter) to individual flows. However, experience with practical systems has revealed the severe scalability problems of the Intserv model due to the amount of routing and reservation state that is required to be maintained in the routers. A natural approach to improving scalability of Intserv architecture is through reduction of amount of state in the routers by using aggregated flow state instead of per-flow state. We present a novel architecture that uses very light state in the routers, while still providing the deterministic guarantees of the Intserv model.**

I. INTRODUCTION

Real-time applications require deterministic delay guarantees, which traditional datagram networks with their window-based end-to-end flow control are ill-equipped to provide. It is now widely accepted that introducing rate-based flow control in the routers is a good approach for providing real-time services. Based on this approach, the Internet community proposed Integrated Services (Intserv [18], [4]) as an architectural framework for providing deterministic guarantees to individual flows in the Internet. In the Intserv model, resources (bandwidth and memory) required by a flow are reserved on a single route from the source to the destination using a signaling protocol (e.g., RSVP [7]) and flows are then serviced at each link using a fair scheduler (e.g., WFQ [2]) to provide the required rate guarantees and enforce isolation between flows.

The Intserv model as currently defined is not scalable for several reasons. Firstly, the per-packet scheduling required at the links can be computationally expensive. For example, the sorted-priority schedulers (e.g., [12], [25], [2], [16], [20]) require $O(log(V))$ time to make a per-packet scheduling decision, where V is the number of connections passing through the link. Therefore, to schedule packets in a timely manner in order to provide delay guarantees, the number of flows through a link must be limited. Fortunately, this problem has been adequately handled by some schedulers that can make scheduling decisions in constant time, irrespective of the number of sessions (e.g., [11], [19]). However, the scalability of the schedulers is severely limited by the amount of state that needs to be maintained by the them. The soft-state approach used in RSVP offers an elegant fault-tolerant mechanism for maintaining reservation state in the routers. However, if the reservation state is high, as is the case when per-flow state is used, then the per-flow periodic refresh messages used by the soft-state mechanism can prove extremely costly. This is especially the case in backbone networks where the number of flows is very high. Accordingly, the second reason why Intserv model is not scalable is that the routers keep state on per-flow basis. Thirdly, the routing state that determines the next-hop for each packet grows proportional to the number of flows through the router. Therefore, if the tables are too large to fit in cache memory, they can be a hindrance in high-speed backbone networks.

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under grants F30602-97-1-0291 and F19628-96-C-0038.

A complete solution to the scalability problem of the Intserv model must address: (1) the per-packet link-scheduling cost (2) the amount of reservation state required to provide guarantees to the flows and (3) the amount of routing state used for forwarding packets towards their destination. Point (1) has been adequately addressed in the literature and we do not pursue further in this paper. A standard approach to state size reduction is to aggregate flows into a small number of flow classes [8] and maintain state on per-flow-class instead of per-flow. Because flows are no longer isolated from each other under flow aggregation, the challenge is to aggregate flows such that providing deterministic guarantees to individual flows is still feasible. In this paper, we suggest a simple technique for flow aggregation, which is flexible with respect to adding and deleting flows from the flow aggregates while maintaining the delay bounds for individual flows.

To date, only partial solutions exist to reducing the routing state of routers. The approaches taken in [21], [22], for example, maintain fixed number of paths to each destination. As the max-flow problem [10] and the minimum-delay problem [23] indicate, under optimal utilization traffic of a particular destination flow towards the destination along an acyclic directed graph or multipath with the destination as the sink node. Taking a clue from this observation, we explore the idea of establishing flows along destination-oriented multipaths instead of using fixed set of paths to destinations. By using one multipath for each destination the routing state can be reduced to levels comparable to those of best-effort architectures. Except for the architecture presented in [15], to the best of our knowledge, there has been no proposal for an architecture that establishes a single flow along multipaths.

Our goal in this paper is to propose a QoS routing architecture that provides delay and bandwidth guarantees to flows and is scalable along all the three dimensions stated above. The proposed architecture combines the scaling advantages of the fixed-path approach with the efficiency of using multipath flows and aggregate-flows to offer a solution that is a reasonable tradeoff between utilization, scalability and quality of delay bounds. The paper is organized as follows. Section II describes in detail our QoS routing architecture. Section III evaluates the complexity of the architecture through analysis and simulations. Section IV concludes this paper.

II. ROUTER ARCHITECTURE

A. Overview

We present our QoS routing architecture in three parts:
1. Flow aggregation
2. Packet forwarding
3. Signaling.

To make Intserv scalable, the per-flow state must be replaced with aggregated state. Aggregating flows removes isolation between flows and, in general, this loss of isolation results in increased burstiness. This can be however controlled if flows are selectively aggregated i.e., only flows that have some common characteristics are allowed in the

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **1999**	2. REPORT TYPE	3. DATES COVERED **00-00-1999 to 00-00-1999**

4. TITLE AND SUBTITLE **A Scalable Architecture for Providing Deterministic Guarantees**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**6**	

aggregation. We introduce the notion of *burst-ratio* and define a small number of aggregated flow classes based on it. Flows belonging to the same class are aggregated and serviced collectively. As a result the link schedulers have to service only a fixed number of queues irrespective of number of flows through the link. The reservation state is reduced to levels where it is feasible to use soft-states for maintenance.

Using a unique routing table entry for each flow as in virtual circuits is a hindrance to scalability and it is highly desirable to use one entry per destination instead. However, establishing flows only along shortest paths is inefficient. We propose *shortest multipath*[23] as a generalization of the shortest path and route flows along the shortest multipath rather than the shortest path. The packets of a single flow can now follow different paths along the multipath to reach the destination. As the results of our simulation experiments show, using multipaths results in higher utilization. Because there is one shortest multipath per destination, the maximum size of the routing table is bounded by the number of nodes in the network rather than the number of flows through the router.

Signaling refers to the process of validating and reserving resources along the QoS path. If the source router of the request determines the complete QoS path using the local link database, the path-selection procedure is said to be *source-directed*. Because the local link database may not have up-to-date information about the non-adjacent links, any selected path has to be validated, and if validation is successful, the necessary resources should be reserved. On the other hand in *hop-by-hop routing*, signaling and path selection proceed in tandem along a fixed multipath. The routers use only the information of adjacent links unlike source-directed routing that requires complete topological information. This has the advantage of not flooding link updates throughout the network. The looping problem of hop-by-hop routing is eliminated by directed-acyclic nature of the shortest multipath. We use this approach here.

B. Flow aggregation

Let a real-time flow be specified by $(\sigma,\ \rho)$ where σ is the maximum burst size and ρ is the average bandwidth required by the flow [1], [16]. We assume that each real-time flow admitted into the network is monitored at the entry point by a token bucket, with bucket size σ and token generation rate ρ, to enforce the flow specification. Define the *burst-ratio* of a flow as the ratio of its burst size to the bandwidth rate, that is, $r = \frac{\sigma}{\rho}$. An equivalent way to specify a flow is $(r,\ \rho)$, where r is the burst-ratio and ρ is the average bandwidth as before, because the burst size can always be obtained by $\sigma = r\rho$. A nice property of flows specified using the burst-ratio is that flows with the same burst-ratio can be merged and divided without changing the burst-ratio of the resulting flows! Let a flow be specified by (σ, ρ), then the amount of traffic A that arrives in an interval $[\tau, t]$ for this flow is given by [16] $A(\tau, t) \leq (\sigma + \rho(t-\tau))$ which is, using burst-ratio $r = \sigma/\rho$, equivalent to

$$A(\tau,t) \quad \leq \quad \rho(r + (t-\tau)) \tag{1}$$

A fluid model is assumed to simplify analysis and focus on the new concepts presented in the architecture. If this flow is split into two streams, A_1 and A_2, such that stream A_1 gets fraction ϕ_1 of the flow and A_2 gets fraction ϕ_2 of the flow, then the arrival pattern for the two output streams is as follows:

$$A_1(\tau,t) \quad \leq \quad \phi_1\rho(r + (t-\tau)) \tag{2}$$
$$A_2(\tau,t) \quad \leq \quad \phi_2\rho(r + (t-\tau)) \tag{3}$$

Therefore, the two resulting streams A_1 and A_2 can be characterized by the parameters $(r, \phi_1\rho)$ and $(r, \phi_2\rho)$, which implies the two subflows have the same burst-ratio. Similarly, let two flows with traffic profiles (r_1, ρ_1) and (r_2, ρ_2), merge into a single flow. The amount of traffic that arrives in an interval $[\tau, t]$ for the aggregated flow is given by

$$A(\tau,t) \quad \leq \quad \rho_1(r_1 + (t-\tau)) + \rho_2(r_2 + (t-\tau)) \tag{4}$$
$$\leq \quad (\rho_1+\rho_2)(\frac{\rho_1 r_1 + \rho_2 r_2}{\rho_1+\rho_2} + (t-\tau)) \tag{5}$$

Accordingly, the aggregate flow can be characterized using burst-ratio by $(\frac{\rho_1 r_1+\rho_2 r_2}{\rho_1+\rho_2}, \rho_1 + \rho_2)$. Note that if $r_1 \leq r_2$, then $r_1 \leq (\frac{\rho_1 r_1+\rho_2 r_2}{\rho_1+\rho_2}) \leq r_2$, which implies the burstiness of merged flow cannot be greater than the burstiness of the more bursty of the two input flows. Therefore, the resulting merged flow can be characterized by $(r_2, \rho_1 + \rho_2)$. The strength of the burst-ratio concept is that characterizing multiplexed and demultiplexed flows becomes tractable; consequently, the delay bounds offered to them by the link schedulers is also simplified.

We now use the burst-ratio to define Q *real-time flow classes*. A burst-ratio R^g is associated with each real-time flow class g, such that $R^{g-1} < R^g$ and R^1 is zero. A flow with specification $(r,\ \rho)$ is classified at the source as belonging to class $g > 1$ if its burst-ratio r is such that $R^{g-1} < r \leq R^g$. Note that there are classes for other non-real-time traffic, such as best-effort; however, we are only concerned with real-time flows in this paper. From Eqs. (1)-(5) it follows that, if two flows belonging to the same class g are merged, then the resulting flow also belongs to the same class g. Similarly, if a flow belonging to a class g is *split* into two or more flows in fixed proportions, then each flow also belongs to the same class g.

A class identifier g is included in every packet that belongs to a flow of class g. The class identifier is used by the schedulers at the links to perform class-oriented fair-scheduling where all packets belonging to the same class are treated as same flow.

Each link in the network is serviced by a fair scheduler like WFQ, which provides bandwidth guarantees and flow isolation. Instead of servicing each individual flow, the schedulers service a fixed number of queues (Q) that correspond to the flow classes. For simplicity we assume that all bandwidth on the link is available for real-time reservation. A variable rb^i_k is associated with each link (i, k), which specifies the unreserved or residual bandwidth. For each real-time class g, let B^g_{ik} be the total bandwidth reserved for real-time class g. Then $rb^i_k = C_{ik} - \sum_{p=1}^{M} B^p_{ik}$ is the residual bandwidth, where C_{ik} is the capacity of the link (i, k). When there is no flow on the link $rb^i_k = C_{ik}$.

A consequence of using a fixed number of flow classes is that the complexity of schedulers at the links is reduced to a constant order. The schedulers now only have to keep the reservation state on an aggregated basis. Only per-flow-class state needs to be maintained, such as the cumulative rate reserved for the class and the time stamps of the last packet belonging to that class. The link schedulers keep no state on a per-flow basis. Packets arriving at the links are aggregated into queues based on the class of the packets, irrespective of their origin and destination. The routing table of a router determines the particular link scheduler the packet enters, and the class label determines the specific queue in the link scheduler that the packet enters. This is far more scalable than architectures in which there is per-flow state. The time complexity of the scheduling decision, as well as the space complexity of the state required to maintain, are now both constant. The admission test for incorporating the resource requirements for a new flow is straightforward — only the availability of the bandwidth on the link and memory in the router need to be checked. The amount of memory for a class g in the scheduler at link (i, k), for lossless delivery, is atmost $R^g B^g_{ik}$ where B^g_{ik} is the bandwidth reserved for class g on the link [16], [1]. In section II-D we describe how the bandwidth and memory reservations are made during flow setup.

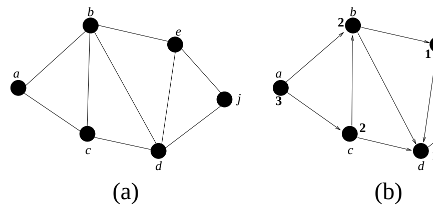


Fig. 1. Successor graph SG_j

C. Destination-oriented Packet Forwarding

Let D^i_j be the distance of i to j measured in number of hops. Define the successor sets as $S^i_j = \{k | k \in N^i \wedge (D^k_j < D^i_j \vee (D^k_j = D^i_j \wedge k < i))\}$. That is, the successor set with respect to a destination consists of all neighbors of a router that are *closer* to the destination and if the neighbor's distance is equal to the router's distance it is included only if the neighbor's address is less than the router's address. Now, with respect to a router j, the successor sets S^i_j define a successor or routing graph $SG_j = \{(m, n) | n \in S^m_j(t),\ m \in N\}$. For example, Fig. 1(b) shows the successor graph SG_j; it can be easily seen that the SG_j is acyclic. A *shortest multipath* [24] from router i to j, is a generalization of shortest path and is defined as the subgraph of SG_j consisting of all nodes that are reachable from the source i. In Fig. 1(b), for example, the links $(b, e), (b, d), (e, j), (e, d), (d, j)$ constitute a multipath from router b to j. Packets received by i destined for j are only forwarded to neighbors in S^i_j. The successor sets can be defined in other ways based on long-term load measurements rather than based on topology, but in this paper we use topology based successor graphs as defined above.

The key idea in our routing technique is that flows for a destination j are only established along the successor graph SG_j. To establish a flow between i and j, any path from i to j in the SG_j that satisfies the QoS requirements can be chosen. Because SG_j is loop-free [23], [24], flows can be established on a hop-by-hop basis by choosing the next router from the successor set at each hop. The signaling procedure is explained in detail in section II-D.

We now describe the routing table structures and the packet forwarding mechanisms. Let $T^i_{g,j}$ be the total rate of the traffic of class g destined for j that router i receives from hosts directly connected to it and from its neighbors. For each $k \in S^i_j$, let the routing parameter $\phi^i_{g,j,k}$ specify the fraction of the traffic $T^i_{g,j}$ that is forwarded to neighbor k. Define the routing parameter set $\Phi^i_{g,j} = \{\phi^i_{g,j,k} | k \in S^i_j\}$. We assume the network does not lose packets. Because the traffic is conserved, the routing parameters satisfy the following properties.

Property 1: For each router i and destination j, the routing parameter set $\Phi^i_{g,j}$ must satisfy the following conditions:

1. For each $\phi \in \Phi^i_{g,j}$, $\phi \geq 0$.
2. $\sum_{\phi \in \Phi^i_{g,j}} \phi = 1$.

A routing table entry is of the form $\langle j, g, S^i_j, \Phi^i_{g,j}, T^i_{g,j} \rangle$. The class g and the destination j uniquely identify a table entry. The forwarding mechanism using the routing tables is described here and the construction of the routing table entry along with signaling is deferred to the next section. Assume each flow is regulated at the ingress router by a token bucket that ensures that the source conforms to its traffic profile. At the ingress router, the source of traffic encodes the destination and class ID in each packet it hands over to the ingress router. Let router i receive a packet belonging to class g and destined for router j. If the destination of the packet is the router itself, the packet is handed over to the higher layer. Otherwise, let the routing table entry corresponding to this destination and class be $\langle j, g, S^i_j, \Phi^i_{g,j}, T^i_{g,j} \rangle$. The first task of the router is to determine a successor k for this packet from S^i_j using the routing parameters $\Phi^i_{g,j}$. This task is carried out by the *distributor* (see Fig. 2) which uses a weighted round robin discipline to allocate packets

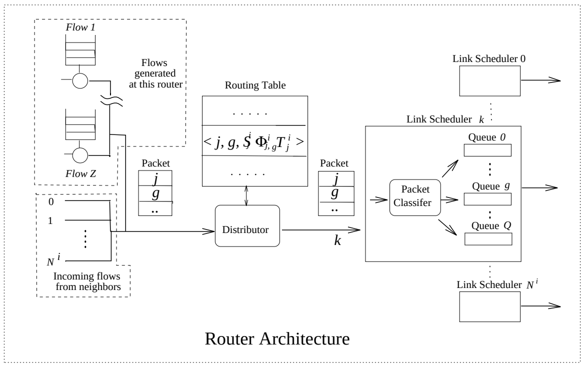


Fig. 2. Routing architecture using multipath packet forwarding

to successors in proportion to the routing parameters. The router then puts the packet in the queue g of the link scheduler of the link to k. The abstract code for packet forwarding is shown in Fig. 3. The time complexity of determining the next hop by the distributor is constant because there are fixed number of neighbors. The packets of a flow can follow any path in the successor graph in a connection-less fashion. There is no explicit connection maintained on per-flow basis. Flows with a particular destination j and class g are aggregated; they collectively share the bandwidth allocated to the class g along the successor graph SG_j.

When flows are added and deleted, the routing tables are modified to reflect the resource requirements. Let a flow request be of class g and bandwidth ρ. Assume the resource provisioning for the flow is to be made along path $P = (n_0, n_1, .., n_m, j)$. Given there is sufficient bandwidth available on the links of the path, for each router $i \neq j$ on the path P and its downstream router k_d, the entry $\langle j, g, S^i_j, \Phi^i_{g,j}, T^i_{g,j} \rangle$ is modified as follows.

$$\phi^i_{g,j,k} \leftarrow \frac{\phi^i_{g,j,k} T^i_{g,j}}{\rho + T^i_{g,j}} \qquad k \neq k_d \tag{6}$$

$$\phi^i_{g,j,k} \leftarrow \frac{\phi^i_{g,j,k} T^i_{g,j} + \rho}{\rho + T^i_{g,j}} \qquad k = k_d \tag{7}$$

$$T^i_{g,j} \leftarrow \rho + T^i_{g,j} \tag{8}$$

If the old routing parameters satisfy the Property 1, then the new routing parameters also satisfy the property. From this, it follows that if bandwidth for all the flows is provisioned in the network and if the flow is token-bucket constrained at the ingress router, the bandwidth requirements of the flows are met.

Flows are aggregated on the basis of their destinations and classes, and the packets of a flow can follow any path along the successor graph after making adequate provisions. The power of this approach lies in that the state in the routers is not increased when a new flow is established. This enables resource provisioning of a single flow along multiple paths in the successor graph. Given a flow with bandwidth ρ it can be divided into m flows with bandwidths ρ_1,..., ρ_m such that $\sum_{i=1}^{m} \rho_i = \rho$. The m flows can then be independently established along different paths P_i in the multipath. What is remarkable is that this does not increase the amount of routing and reservation state in the routers. Later, we show through simulations how multipath flows can reduce bandwidth fragmentation and improve call-acceptance rates.

Figure 2 shows our router architecture. In the figure, router i has N^i links and each outgoing link is serviced by a WFQ scheduler. Router i uses the token buckets to enforce the rates of the Z flows generated locally by the host attached to the router; however, the incoming flows on the links need no such enforcement. When the router receives a

```
procedure $PacketForwarding(M)$
{Executed at router $i$. M is a data message.}
begin
    let $M = (j, g, data)$; // $j$ is dest, $g$ is class, rest is data
    let $E = \langle j, g, S^i_j, \Phi^i_{g,j}, T^i_{g,j} \rangle$ be the table entry for the packet;
    let $k$ be the neighbor that the distributor determined
        based on the routing parameters in $\Phi^i_{g,j}$;
    Enqueue the packet $M$ in queue $g$ of link scheduler to $k$ ;
end
```

Fig. 3. Packet Forwarding Procedure

packet with destination j and class g, the corresponding routing table entry is accessed. The packet is then added to the queue g of the WFQ of the link to neighbor k determined by the distributor.

D. Signaling

A flow request of the form $(j,\ g,\ \rho)$ is made by an application to the source router, where j is the destination and ρ is the bandwidth of the flow and g is the class of the flow. The source router typically selects a valid path *along* the successor graph SG_j that satisfies the QoS requirements by running a path-selection algorithm on the topology information in its database. It then initiates hop-by-hop signaling to reserve resources along the determined paths. Source-directed signaling requires full topology and current utilization information of each link. Alternatively, the source can initiate a flow setup on a hop-by-hop basis in which path-selection and signaling proceed in tandem; this requires only information regarding adjacent links. Along another dimension, the signaling procedure can be classified as being either an *end-to-end* type or *in-line* type. In end-to-end signaling, a flow is first established before data is transmitted, while in the in-line signaling data packets immediately follow the setup messages. The particular type of signaling is not important for the material presented in this paper; however, to simplify our presentation we use the hop-by-hop end-to-end signaling model in our descriptions. The techniques presented in this paper can be easily adapted to any other type of signaling.

Signaling process has an admission-test phase and a commit phase. During the admission-test phase the availability of bandwidth on each link on the path is tested; if the test passes at each hop on the path, only then the resources are committed. The abstract code for the flow setup is shown in Figure 4. A signaling message is of the form $REQ = (j, g, \rho)$, where j is the destination, g is the class of the flow and ρ is the bandwidth that needs to be reserved. A reply message is of the form $RPY = (s)$, where s specifies the status and can have one of the two values SUCCESS or FAILURE. The signaling process is initiated by the source n_0 where the request originated. When the destination receives the request it simply replies SUCCESS. When an intermediate router i, receives a flow setup message from upstream router k_u, it uses a heuristic to determine $k_d \in S^i_j$ that offers the best choice for reserving the bandwidth for the flow. If no such successor exists, the router simply returns a failure to the upstream router; otherwise, the bandwidth ρ is blocked on the link to k_d and the request is forwarded to k_d. The router then waits for a reply from k_d. If the reply is a FAILURE, then a FAILURE status is returned to the upstream router k_u. Conversely, if the reply is a SUCCESS, the router commits the bandwidth on the link and returns a SUCCESS status to the upstream router. At this time it updates the routing parameters in the routing table using the Eqs. (6)-(8). The signaling process ends when the initiator n_0 receives a reply $RPY = (s_0)$. If s_0 is SUCCESS, then the session at n_0 that initiated the flow setup is informed about the status, which can then start sending data.

```
procedure $FlowSetup(REQ, k_u)$
{REQ is a setup message received from neighbor $k_u$}
begin
    let $REQ = (j, g, \rho)$;
    Select $k_d \in S^i_j$ for which the residual bandwidth($rb$) is largest;
    if ( $\rho > rb^i_{k_d}$ ) then
        Send reply RPY=$FAILURE$ to $k_u$;
    else
        Block bandwidth $\rho$ on link $(i, k_d)$;
        Send message $REQ$ to $k_d$;
    endif
    Wait for reply from $k_d$;
    let the reply be $RPY = (s)$;
    if ($s = FAILURE$) then
        $rb^i_{k_d} \leftarrow rb^i_{k_d} + b$; // release bandwidth $b$ on link $(i, k_d)$
        Send reply RPY=($FAILURE$) to $k_u$;
    else
        Adjust routing parameters using Eqs. (6)-(8);
        Send $RPY = (SUCCESS)$ to $k_u$;
    endif
end
```

Fig. 4. Flow setup using Hop-by-hop signaling

```
procedure $FlowRelease(REL)$
{Executed at router $i$. REL is a release message.}
begin
    let $REL = (j, g, \rho)$;// $j$ is label, $g$ is class, $\rho$ is bandwidth
    if ($j = i$) then // destination is reached
        return ;
    else
        Find $\rho_k$ for each $k \in S^i_j$ such that $\sum \rho_k = \rho$;
        $rb^i_k \leftarrow rb^i_k + \rho,\ \forall k \in S^i_j$,;
        Recompute routing parameters using Eqs. (9)-(10);
        For each $k \in S^i_j$, Send $REL_k = (j, g, \rho_k)$ to $k$;
    endif
end
```

Fig. 5. Flow teardown algorithm

The heuristic for choosing the next hop from the successor set can use any strategy. For example, the next hop k chosen from S^i_j may be the widest outgoing link in the successor set, i.e., the successor link with largest rb^i_k. Another heuristic consists of using the utilization of the link as a metric. That is, choose the neighbor k from the successor set for which α/rb^i_k is the lowest, where rb^i_k is the residual bandwidth on the link (i, k) and α is some constant. In this paper, we choose the widest-outgoing-link heuristic for simulation purposes.

When a session is terminated, the ingress router initiates a flow teardown procedure. The flow teardown or release request is of the form $REL = (j, g, \rho)$, where j is the destination of the flow, ρ is the bandwidth of the session and g is the class of the flow. The abstract code for flow release is shown in Fig. 5. Let router $i \neq j$ receive $REL = (j, g, \rho)$. A heuristic is used to determine, for each $k \in S^i_j$, ρ_k such that $\sum \rho_k = \rho$. Because a flow teardown is performed only after the flow establishment, the heuristic can find such a distribution. For each $k \in S^i_j$, the bandwidth ρ_k is released from the link to k and the following equations are used to update the routing table.

$$\phi^i_{g,j,k} \quad \leftarrow \quad \frac{\phi^i_{g,j,k} T^i_{g,j} - \rho_k}{T^i_{g,j} - \rho} \quad k \in S^i_j \tag{9}$$

$$T^i_{g,j} \quad \leftarrow \quad T^i_{g,j} - \rho \tag{10}$$

Then, for each $k \in S^i_j$, a release request $REL_k = (j, g, \rho_k)$ is forwarded to router k. The same steps are repeated at the receiving routers. The process continues until when the release message reaches the destination. The destination simply discards a release message. If the old routing parameters satisfy the Property 1, the new routing parameters also satisfy Property 1.

III. Analysis and Simulations

Delay and bandwidth guarantees: The delay bound offered to a flow with specification (σ, ρ) by a WFQ at a link with capacity C and maximum packet size L is given by $\frac{\sigma}{\rho} + \frac{L}{\rho} + \frac{L}{C}$ [16] which reduces to the burst-ratio of the flow $r = \frac{\sigma}{\rho}$, if a fluid model (i.e., $L = 0$) is assumed. Therefore, the delay bound offered to flow of class g at *any* link is R^g. The maximum delay for a single-path flow can be easily obtained from just the path length measured in terms of hops and the class of the flow. Let $max(P_{ij})$ be the length of the longest path, in terms of hops, from i to j in the successor graph SG_j. If the flow belongs to class g, then the maximum delay for any flow from i to j is $max(P_{ij}) \times R^g$. Because the successor graphs are precomputed using the distances to destinations, the delay bounds can be derived by the user from the specification of the flow alone!

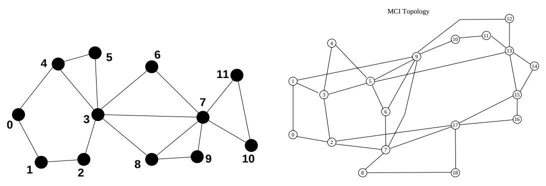
The bandwidth guarantees follow from the fact that bandwidth for each flow is provisioned by the flow setup protocol, each link is serviced by the WFQ to provide the minimum bandwidth on the flow path and the routing parameters always satisfy Property 1.

The delay analysis under non-fluid model is more complex and is beyond the scope of this paper. Under a non-fluid model, the distributors and the link schedulers introduce some jitter or delay variation that must be eliminated using traffic shapers at each hop.

Time Complexities: The packet scheduling time complexity is $O(log(Q))$ if a sorted-priority scheduler is used. The time complexity of the distributor component of the router is $O(N^i)$.

Space Complexities: The reservation state size is of order $O(Q)$. Aggregation based on burst-ratio is very powerful in that a large number of flows can be aggregated into few queues. The aggregation scheme suggested here depends only on the available bandwidth on the link, and is independent of reservations already made to the existing flows, which is unlike the scheme used in [8], for example. The size of the routing table is of order $O(Q|N|)$. It is reasonable to assume that $|N|$ is much larger than Q, which makes the size of the routing table of the same order as in best-effort architectures. The size of a routing-table entry is larger compared to best-effort architecture, because of the extra information needed to provide the guarantees, but the entry is of a fixed size determined by the number of neighbors of the router.

Effect of Multipath Flows on Call-blocking Rates: Call-blocking rate is a metric used to measure the performance of the path-selection algorithms. Call-blocking rate is defined as the percentage of requests that are rejected by the network. Prior studies [13], [17], [6] have used call-blocking rates to evaluate various path-selection algorithms and the link update policies. When large number of high-bandwidth requests are issued, they tend to cause bandwidth fragmentation in the network which increases the overall call-blocking rate. Bandwidth in the network is said to be fragmented if a flow is rejected because there is no single path that provides the requested bandwidth, but there are two or more non-identical paths that collectively provide the bandwidth. We show through simulations that establishing a single flow along multiple paths can reduce fragmentation of bandwidth in the network and increase overall call-acceptance rate. Because none of the current architectures support multipath flows, all path-selection algorithms studied to date use a single-path for a flow. We use the following schemes to show the effect of multipath flows on call-blocking rate.



(a) CAIRN topology (b) MCI topology

Fig. 6. Topologies used in simulations

- S Scheme: The bandwidth requirement of the requested flow is signaled along a single path in the network. The widest (i.e., largest residual bandwidth) outgoing link in the successor set is chosen at each hop during the signaling process.
- 'Mn' Scheme: The flow request is first divided into n small flows of equal size and each of these n flows is independently setup using the S scheme.
- 'WS' Scheme: Among the feasible paths, the path that has the shortest length is chosen and, if more than one such path is available the one that offers the widest bandwidth is chosen [13].

For simulations, we use the internet topologies shown CAIRN and MCI in Fig. 6; the topologies shown differ from the real networks in the capacities. Identical flow requests are generated and signaled using the S and Mn schemes and the blocking rate is measured. Flow requests have a uniform distribution across the network; that is, the source and destination are randomly chosen with uniform distribution, and the bandwidth is chosen uniformly within certain bandwidth range. Here, we are interested in comparing only the performance effects of using multipath flows, so for simplicity, we assume that link-state information propagates instantaneously throughout the network, that is, each router has the most current information regarding the links. Also the flow signaling takes zero time, and each flow has infinite duration. The effects of variations in arrival pattern of the requests is not studied here.

Observe that, when the flow requests are of small size (0.3 to 0.4 Mb) in Figs. (7) and (8), the performance of M4 and M8 is only slightly better than S, i.e., the improvements in call-acceptance achieved through Mn schemes are minimal when flow requests have relatively small bandwidths. However, the benefits of using multipaths are pronounced when flow requests are of larger bandwidths. When the requests have bandwidths in the range 3-4 Mb, the call-acceptance rates are as shown in Figs. (9) and (10). As the number of flows into which the original flow is partitioned increase, the call acceptance rates also increase. The basic result is that the throughput of the network increases when flows are broken down into small flows due to the better use of network bandwidth. If a QoS architecture supports multipath flows without increasing state in the network, like the one presented in this paper, the blocking rates can be further improved by using multiple path flows.

Figs. (7)-(10) also show curves for the ideal widest-shortest path (WS) scheme. The performance of the WS scheme is better than M4 and M8 because it uses complete topology information to optimize, unlike the hop-by-hop approach in this paper that uses only information of the adjacent links. Since non-adjacent link information may not be correct, link information is periodically flooded throughout the network. This extra performance of WS comes at the cost of the periodic updates. In the particular signaling used here routers need not know the link utilization information of non-adjacent links. Furthermore, the performance of WS scheme shown is 'ideal', in the sense that routers have instantaneous access to the latest link information which is not true in practice. The performance of WS in practice would be worse than what is shown in the figures. However, when large requests are

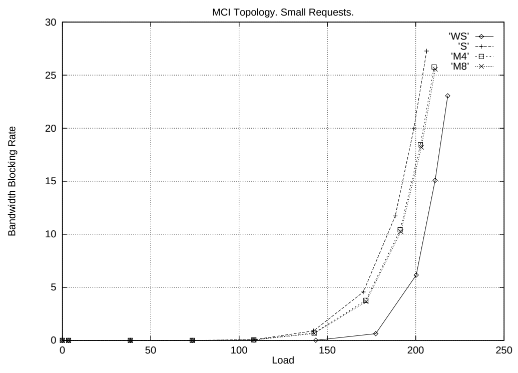


Fig. 7. Bandwidth Blocking rate for small requests under Hop Routing. MCI topology.

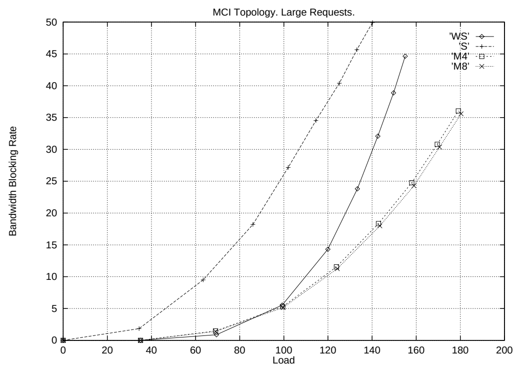


Fig. 9. Bandwidth Blocking rate for large requests under Hop Routing. MCI topology.

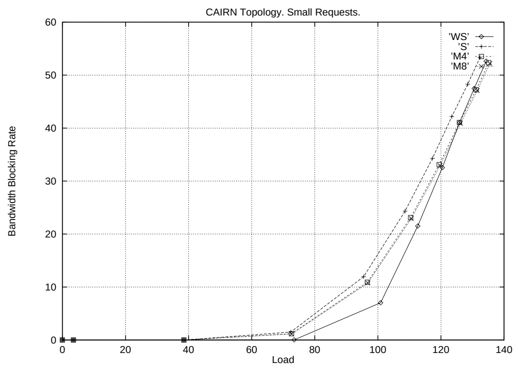


Fig. 8. Bandwidth Blocking rate for small requests under Hop Routing. CAIRN topology.

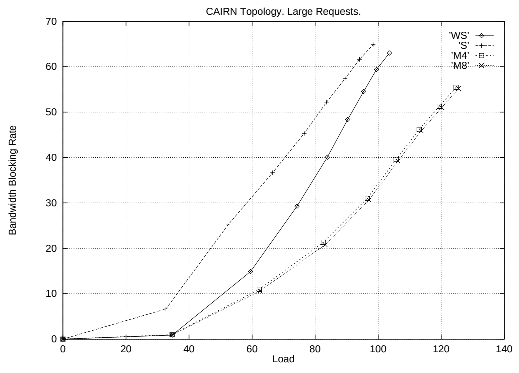


Fig. 10. Bandwidth Blocking rate for large requests under Hop Routing. CAIRN topology.

involved the performance of our signaling is better than the ideal WS scheme. Our approach strikes a reasonable balance between complexity of the architecture and bandwidth utilization. The simulations we present here are quite preliminary and are intended to illustrate that the proposed scheme achieves significant routing state aggregation and network utilization.

IV. Conclusion

A scalable QoS architecture has been presented that uses highly aggregated state in the routers, and yet provides deterministic delay and bandwidth guarantees. We introduced the concept of burst-ratio to aggregate flows into a fixed number of flow classes. The burst-ratio technique is simple, flexible and powerful, as it enables aggregation of a large number of flows into a small number of fixed queues, thereby reducing the time and storage complexity of the link schedulers to a constant. The flows are routed along fixed multipaths and as a result the architecture provides multiple paths to destinations while having routing table size complexity of $O(QN)$, where Q is the number of classes and N is number of routers in the network. Overall, the need for maintaining per-flow state in the routers is eliminated, making the routing architecture and the associated signaling protocols simple and scalable.

References

[1] R.L. Cruz. A calculus for network delay, Part I: Network Elements in isolation. *IEEE Trans. Inform. Theory*, 37(1):114–121, 1991.

[2] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. *Proc. of ACM SIGCOMM*, pages 1–12, 1989.

[3] D. Black et al. An Achitecture for Differentiated Services. *Internet Draft*, May 1998.

[4] E. Crawley et al. A framework for qos-based routing in the internet. *Internet Draft*, April 1998.

[5] G. Apostolopoulos et al. On reducing the processing cost of on-demand QoS path computation. *Journal of High Speed Networks*, 7:77–98, 1998.

[6] G. Apostolopoulos et al. Quality of Serive Based Routing: A Performance Perspective. *Proc. of ACM SIGCOMM*, 1998.

[7] L. Zhang et al. RSVP: A New Resource Reservation Protocol. *IEEE Communications Magazine*, 31(9):8–18, 1993.

[8] R. Guerin et al. Scalable QoS Provision Through Buffer Management. *Proc. of ACM SIGCOMM*, 1998.

[9] Y. Bernet et al. An Framework for Differentiated Services. *Internet Draft*, May 1998.

[10] L.R. Ford and D.R. Fulkerson. *Flow in Networks*. Princeton University Press, Princeton, New Jersey, 1962.

[11] S.J. Golestani. A Framing Strategy for Congestion Management. *IEEE Journal on Selected Areas in Communications*, pages 1064–1077, Sept. 1991.

[12] S.J. Golestani. A Self-Clocked Fair Queueing Scheme for Broadband Applications. *Proc. IEEE INFOCOM*, pages 636–646, 1994.

[13] Q. Ma and P. Steenkiste. On path selection for traffic with bandwidth guarantees. *Proc. International Conference on Network Protocols*, October 1997.

[14] Q. Ma and P. Steenkiste. Quality-of-service routing for traffic with performance guarantees. *Proc. IFIP International Workshop on Quality of Service*, pages 115–126, May 1997.

[15] S. Murthy and J.J. Garcia-Luna-Aceves. Congestion-oriented shortest multipath routing. *Proc. IEEE INFOCOM*, March 1996.

[16] A.K. Parekh and R.G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Trans. Networking*, 1:344–357, June 1993.

[17] A. Shaikh, J. Rexford, and K. Shin. Efficient computation of Quality-of-Service routes. *NOSSDAV*, 1998.

[18] S. Shenker, D. Clark, and L. Zhang. A service model for an integrated services internet. *Internet Draft*, Oct. 1993.

[19] M. Shreedhar and G. Varghese. Efficient Fair Queueing using Deficit Round Robin. *Proc. of ACM SIGCOMM*, 1995.

[20] D. Stiliadis and A. Varma. Rate-Proportional Services: A Design Methodology for Fair Queuing Algorithms. *IEEE/ACM Trans. Networking*, April 1998.

[21] I. Stoica and H. Zhang. LIRA: A model for service differentiation in the Internet. *NOSSDAV*, July 1998.

[22] I. Stoica and H. Zhang. Providing Guaranteed Services Without Per Flow Management. *Proc. of ACM SIGCOMM*, Sept. 1999.

[23] S. Vutukury and J.J. Garcia-Luna-Aceves. A Simple Approximation to Minimum Delay Routing. *Proc. of ACM SIGCOMM*, 1999.

[24] W. T. Zaumen and J.J. Garcia-Luna-Aceves. Loop-Free Multipath Routing Using Generalized Diffusing Computations. *Proc. IEEE INFOCOM*, March 1998.

[25] L. Zhang. Virtualclock: A New Traffic Control Algorithm for Packet Switching Networks. *Proc. of ACM SIGCOMM*, pages 19–29, 1990.